



^{*a*}Department of Computer Science, Cornell University, Ithaca NY ^{*b*}CASC, Lawrence Livermore National Labs, Livermore CA

When compiling or analyzing a program, the first step is the conversion of the source code into an Intermediate Representation (IR) which is organized in an Abstract Syntax Tree (AST). The AST captures all the information in the source code in a form suitable for analysis. However, the size of the AST is proportional to the number of lines of code, which makes storing the AST in memory infeasible for large projects. To combat this problem, we utilize the ROSE framework to “compress” ASTs by performing a merge operation to combine IR nodes that represent the same declaration (e.g. declarations in a header file included from multiple locations). Doing so allows us to reduce the amount of space required by an AST without losing any information required for compilation or analysis.

- ROSE is a framework for building source to source translators
- Translator converts source code into an *Intermediate Representation* (IR), with nodes representing individual source constructs.
- IR nodes are linked together into an *Abstract Syntax Tree* (AST) representing the source code in a heirarchical manner.
 - AST used for optimization and analysis.
- Number of IR nodes (hence, AST size) directly proportional to code size.
- For large programs (such as many lab projects), storage requirements for AST are prohibitive.
- Many IR nodes are duplicates (e.g. duplicate declarations from header files)
- Rather than having duplicate IR nodes, we can have a single IR node and reference it from multiple locations

Source code



- For every declaration in AST, generate a unique identifier
 - Akin to name mangling in compilers
- Build a mapping from unique identifiers to IR nodes
 - Identifier maps to the IR node of the *first declaration having that identifier* (the *reference* node).
- For every link in AST to the IR node of a declaration, use mapping to re-link to reference node.
- Remove all duplicate IR nodes.



- ROSE provides support, through a *rewrite mechanism*, for inserting new code into the AST
 - Useful for performing source-to-source compiler optimizations
- ROSE builds *secondary AST* for code fragment to be inserted, then inserts code fragment into primary AST.
- Any links in code fragment to declarations initially link to secondary AST, necessitating its saving.
- Use merging mechanism to re-link the code fragment to declarations in primary AST.
- Now secondary AST can be removed safely.

- Can eliminate all redundant declarations in AST
 - Especially useful for commonly included standard headers, such as `iostream` – each header file only represented once in AST.
- Namespaces with identical names but different contents are partially merged.
- Merging does not affect proper operation.
 - All test codes – including uses of rewrite mechanism – succeed
- AST sizes now roughly proportional to *unique* lines of code, instead of total lines of code
- Future work: Leverage merging mechanism to perform analyses on large laboratory projects.

UCRL